Heuristic Optimisation

Part 8: Simulated annealing

Sándor Zoltán Németh

http://web.mat.bham.ac.uk/S.Z.Nemeth

s.nemeth@bham.ac.uk

University of Birmingham

Overview

- Introduction
- Basic structure of Simulated Annealing
- Properties of Simulated Annealing
- SA applied to SAT, TSP, NLP

Introduction

In the early 80's Kirkpatrick, Gelatt, and Vecchi and independently Cerny introduced the concepts of annealing in combinatorial optimisation.

In condensed matter physics annealing is known as a thermal process for obtaining low energy states of a solid in a heat bath.

The concept of Simulated Annealing is based on a strong analogy between the physical annealing process of solids and the problem of solving large combinatorial optimisation problems.

Physical annealing

In order to reach a low energy state:

1. The temperature of the heat bath is increased to a maximum value at which the solid melts.

   In this liquid phase the particles of the solid arrange themselves randomly.

2. The temperature of the heat bath is carefully decreased until the particles arrange themselves in the ground state of the solid.

Physical annealing cont'd

The goal of the physical annealing process is to reach the ground state of the solid where the particles are arranged in a highly structured lattice and the energy of the system is minimal.

The ground state of the solid is obtained only if the maximum temperature is sufficiently high and the cooling is done sufficiently slowly.

Otherwise the solid will be frozen into a meta-stable state rather than into the ground state.

Analogy

Methods for simulation of the physical annealing process can be directly applied to solve optimisation problems.

Physical system	Optimisation problem
state	feasible solution
energy	evaluation function value
ground state	optimal solution
temperature	control parameter T
careful annealing	simulated annealing

Iterated hill-climbing

```
1. Choose a starting solution $i_{start}$ and initialise $best$
2. $k := 0; i := i_{start}$
3. Repeat until $k = MAX$:
   (a) Repeat until local optimum is found:
       i.  select $j$ as the neighbour of $i$ with best value of evaluation function $F(j)$
       ii. if $F(j)$ is better than $F(i)$ then $i := j$
           else local optimum found
   (b) $k := k + 1$
   (c) if $j$ is better than $best$ then $best := j$
   (d) $i :=$new random solution
```

Modification to iterated hill-climbing

Instead of checking all neighbours of a current point *i*, select just one point *j* from the neighbourhood

Accept the new point with some probability depending on the relative merit of the two points *i* and *j*

Stochastic hill-climbing

Maximisation:

1. Choose a starting solution i_{start} and evaluate it
2. $k := 0; i := i_{start}$
3. Repeat until $k = MAX$:
    - (a) select j as a neighbour of i
    - (b) $i := j$ with probability $\frac{1}{1+e^{\frac{F(i)-F(j)}{T}}}$
    - (c) $k := k + 1$

Properties of stochastic hill-climbing

For $F(i) = 107$, $F(j) = 120$:

- if $T = 1$ the probability of acceptance is close to 1
- if $T = 10^{10}$ the probability of acceptance is 0.5

For $T = 10$ and $F(i) = 107$:

- if $F(j) = 80$ the probability of acceptance is 0.06
- if $F(j) = 100$ the probability of acceptance is 0.33
- what if $F(j) = F(i)$?
- if $F(j) = 150$ the probability of acceptance is 0.99

Basic structure of SA algorithm

```
1. Choose a starting solution $i_{start}$
2. Initialize $T_0$,$M_0$
3. $k := 0; i := i_{start}$
4. Repeat until halting criterion is satisfied:
    (a) Repeat $M_k$ times:
        i. generate $j$ as a neighbour of $i$
        ii. if $F(j) isbetterthanF(i)$ then $i := j$
            else if $e^{-\frac{|F(i)-F(j)|}{T_k}} > random(0,1)$ then $i := j$
    (b) $k := k + 1$
    (c) Calculate $M_k$
    (d) Calculate temperature $T_k$
```

Properties of SA

For maximisation the inequality in step (a)ii is equivalent to $F(j) > F(i) + T_k \log(random(0,1))$.

- In contrast to hill-climbing, simulated annealing accepts some deterioration in the quality of solutions. This helps avoiding local optima.
- Initially, at high temperatures, large deteriorations are accepted.
- As temperature decreases, only smaller deteriorations are accepted.
- As temperature approaches 0, SA behaves as local optimisation.
- Simulated annealing is a generalisation of local search.

Difficulties of SA

- Problem specific questions:
  What is a solution?

  What are the neighbours of a solution?

  What is the value of a solution?

  How do we determine the initial solution?

- Adjusting the control parameters:
  How do we initialise T_k and M_k?

  How do we determine cooling (get next value for T_k)?

  How do we calculate M_k in each step?

  What should be the halting criterion?

SA for SAT

```
1. Repeat steps 2.-4. MAX_TRIES times:
2. Assign values to $X =< x_1, \dots, x_n >$ randomly
3. $k := 0$
4. Repeat until $T_k < T_{min}$
     If the formula is satisfied, return X
     else
           $T_k := T_{max} \times e^{-kr}$
           compute the increase in the number
           of satisfied clauses $\delta$, if $x_i$ was flipped
           flip $x_i$ with probability $(1 + e^{-\frac{\delta}{T_k}})^{-1}$
     $k := k + 1$
5. Return “No solution found”
```

SA for TSP

The basic SA algorithm can be used.

Differences between implementations:

- the methods of generating the initial solution
- the definition of a neighbourhood for a given tour
- the selection of a neighbour
- the methods for decreasing temperature
- the halting condition
- possible postprocessing

SA for NLP

Neighbourhood can be defined using a Gaussian distribution for each variable:

$$\begin{aligned} \mathbf{x} &= (x_1, \ldots, x_n),\ l_i \le x_i \le u_i \\ x_i' &\leftarrow x_i + N(0, \sigma_i), \end{aligned}$$

$N(0, \sigma_i)$ being and independent random Gaussian number with mean 0 and standard deviation $\sigma_i = \frac{u_i - l_i}{6}$

For the maximisation of G_2,
$\mathbf{x}'$ is definitely accepted if $G_2(\mathbf{x}') > G_2(\mathbf{x})$

otherwise with probability $e^{\frac{G_2(\mathbf{x}') - G_2(\mathbf{x})}{T}}$

If the probability of acceptance is P, then
$G_2(\mathbf{x}') = G_2(\mathbf{x}) + T \log P$.

Recommended reading

Z. Michalewicz & D.B. Fogel
How to Solve It: Modern Heuristics

Chapter 5. Escaping Local Optima

Section 5.1 Simulated Annealing